09/995,925

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L2	8	(US-20020078189-$ or US-20030100968-$).did. or (US-4240137-$ or US-4677587-$ or US-5313615-$ or US-5325526-$ or US-5630135-$ or US-5638522-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/27 12:41
L3	2	l2 and (((new or different) or updat$2) near2 input)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 12:43
L4	3	(execut$4) near3 ((block near3 program) same (bas$2 near2 (updat$2 or new (input)))) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:42
L5	0	(execut$4) near3 ((block near3 program) same (bas$2 near2 ((updat$2 or new) near2 input))) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:43
L6	0	(execut$4) near6 ((block near3 program) same (bas$2 near2 ((updat$2 or new) near2 input))) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:44
L7	0	(execut$4) near6 ((block near3 program) same (bas$2 near4 ((updat$2 or new) near2 input))) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:46
L8	0	((block near3 program) same (bas$2 near4 ((updat$2 or new) near2 input))) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:47
L9	69	(block near3 program) same ((updat$2 or new) near2 input) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:51

L10	14	(execut$4 or test) same ((block near3 program) same ((updat$2 or new) near2 input)) and (@ad<"20011128" or @rlad<"20011128" or @prad<"20011128")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/27 13:48

Terms used **execute block program only new input** Found **117,220** of **157,956**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

2 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science

William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, David M. Young

March 1968 **Communications of the ACM**, Volume 11 Issue 3

Full text available: pdf(6.63 MB) Additional Information: full citation, references, citings

Keywords: computer science academic programs, computer science bibliographies, computer science courses, computer science curriculum, computer science education, computer science graduate programs, computer science undergraduate programs

3 Secure program execution via dynamic information flow tracking

G. Edward Suh, Jae W. Lee, David Zhang, Srinivas Devadas

October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**, Volume 39 , 38 , 32 Issue 11 , 5 , 5

Full text available: pdf(263.33 KB) Additional Information: full citation, abstract, references, index terms

We present a simple architectural mechanism called dynamic information flow tracking that can significantly improve the security of computing systems with negligible performance overhead. Dynamic information flow tracking protects programs against malicious software